In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

EDA

In [2]:
```
df_train= pd.read_csv('train.csv')
```

In [3]:
```
df_train.head()
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabi
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	Na
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C8
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	Na
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C12
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	Na

Questions to ask: 1. Who were the passengers(Age,Sex,Pclass) - Demography analysis (2) 2. Attribute wise survival Probability (3) 3. Survival Probability Distribution (1)

In [4]:
```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

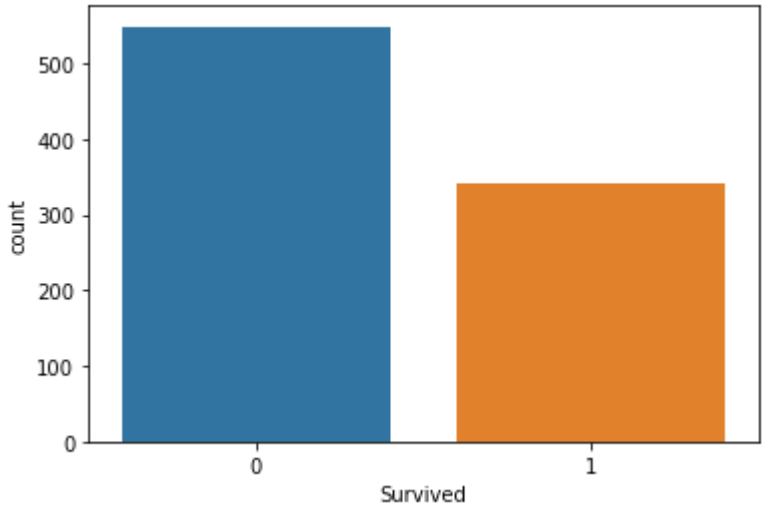
In [5]:
```
df_train.isna().sum()/df_train.shape[0]

## Percentage of Null Values in each column
## While modelling if any column has greater than 25-30% null values you need to
```

Out[5]:
```
PassengerId    0.000000
Survived       0.000000
Pclass         0.000000
Name           0.000000
Sex            0.000000
Age            0.198653
SibSp          0.000000
Parch          0.000000
Ticket         0.000000
Fare           0.000000
Cabin          0.771044
Embarked       0.002245
dtype: float64
```

In [6]:
```
sns.countplot(x='Survived',data=df_train)
```

Out[6]:
```
<AxesSubplot:xlabel='Survived', ylabel='count'>
```



In [7]:
```
100*df_train['Survived'].value_counts()[1]/df_train.shape[0]

## total percentage of people who survived
```

Out[7]:
```
38.38383838383838
```

Demography Analysis

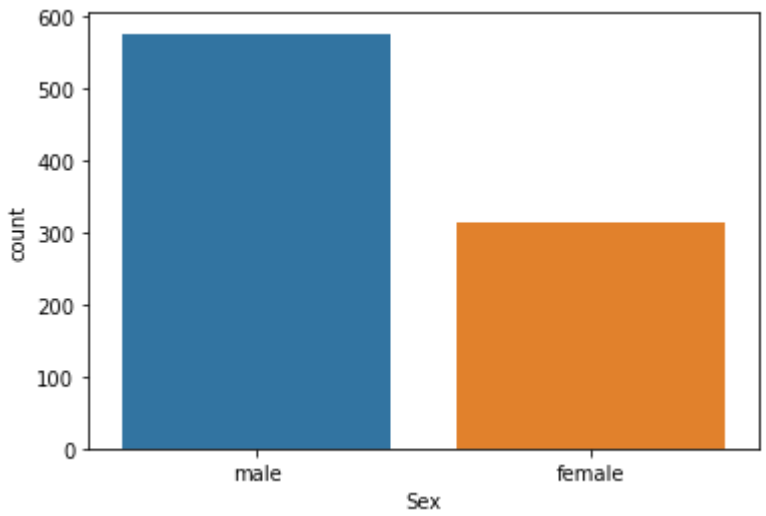
In [8]:
```
df_train.head()
```

Out[8]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabi
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	Na
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C8
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	Na
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C12
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	Na

In [9]:
```
sns.countplot(x='Sex',data=df_train)
```

Out[9]: <AxesSubplot:xlabel='Sex', ylabel='count'>

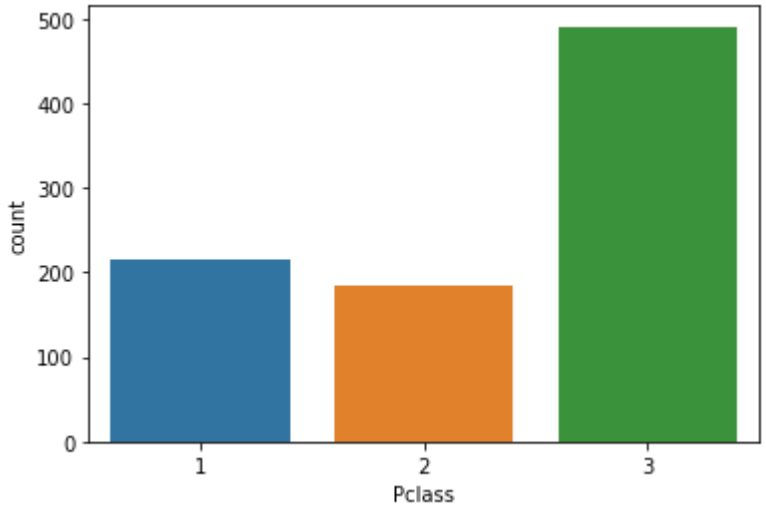


In [10]:
```
100*df_train['Sex'].value_counts()['female']/df_train.shape[0]
## total percentage of people who survived
```

Out[10]: 35.24130190796858

In [11]:
```
sns.countplot(x='Pclass',data=df_train)
```

Out[11]: <AxesSubplot:xlabel='Pclass', ylabel='count'>

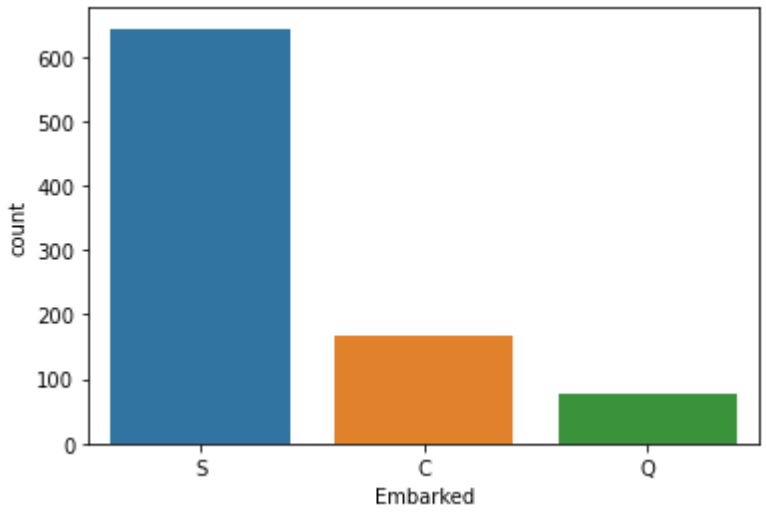


In [12]:
```
for i in df_train['Pclass'].unique():
    print(f"For Pclass{i}:{100*df_train['Pclass'].value_counts()[i]/df_train.shap
```

```
For Pclass3:55.10662177328844
For Pclass1:24.242424242424242
For Pclass2:20.650953984287316
```

In [13]:
```
sns.countplot(x='Embarked',data=df_train)
```

Out[13]: <AxesSubplot:xlabel='Embarked', ylabel='count'>



In [14]:
```
for i in df_train['Embarked'].dropna().unique():
    print(f"For Embarked{i}:{100*df_train['Embarked'].value_counts()[i]/df_train.
```

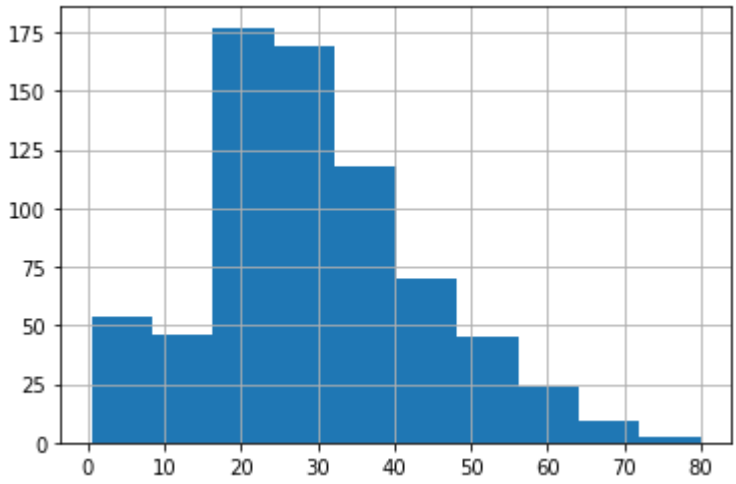
```
For EmbarkedS:72.27833894500561
For EmbarkedC:18.855218855218855
For EmbarkedQ:8.641975308641975
```

In [15]:
```
df_train['Embarked'].unique()
```

Out[15]:
```
array(['S', 'C', 'Q', nan], dtype=object)
```

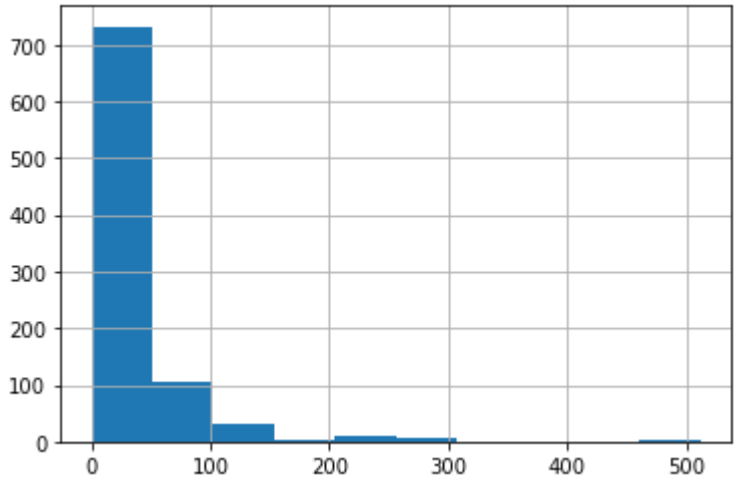
In [16]:
```
df_train['Age'].hist()
```

Out[16]:
```
<AxesSubplot:>
```



In [17]:
```
df_train['Fare'].hist()
```

Out[17]:
```
<AxesSubplot:>
```



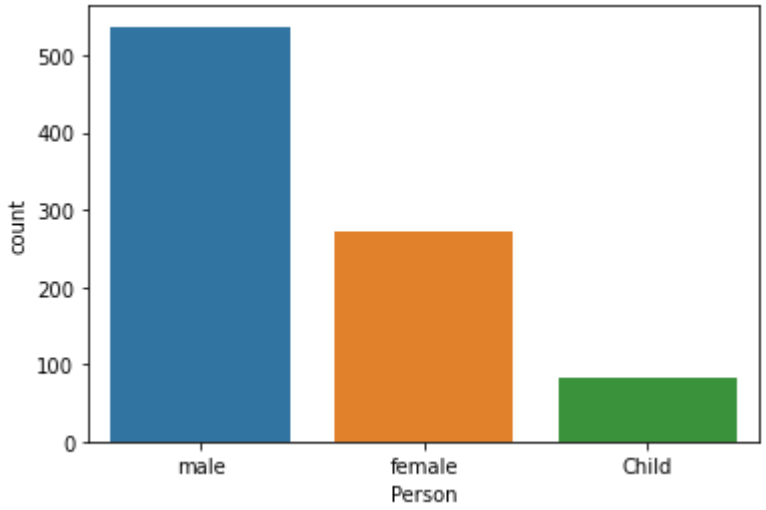
Age and Sex -> New Variable

In [18]:
```
def Person(x):
    Sex,Age=x
    if Age<16:
        return "Child"
    else:
        return Sex
```

In [19]:
```
df_train['Person']=df_train[['Sex','Age']].apply(Person,axis=1)
```

In [20]:
```
sns.countplot(x='Person',data=df_train)
```

Out[20]: <AxesSubplot:xlabel='Person', ylabel='count'>



In [21]:
```
for i in df_train['Person'].dropna().unique():
    print(f"For a Person being{i}:{100*df_train['Person'].value_counts()[i]/df_tr
```

```
For a Person beingmale:60.26936026936027
For a Person beingfemale:30.41526374859708
For a Person beingChild:9.31537598204265
```

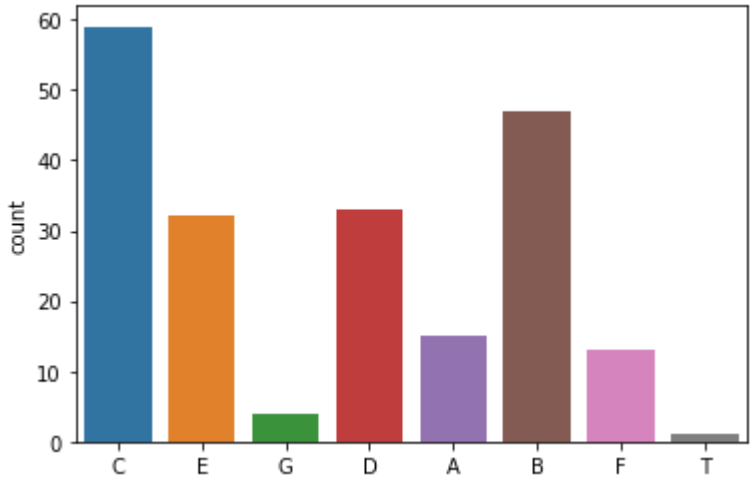
In [22]:
```
## data cleaning to analyse cabin
x=list(df_train['Cabin'].dropna())
```

In [23]:
```
cabin=[]
for i in x:
    cabin.append(i[0])
```

In [24]:
```
sns.countplot(pd.Series(cabin))
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarnin
g: Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without a
n explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

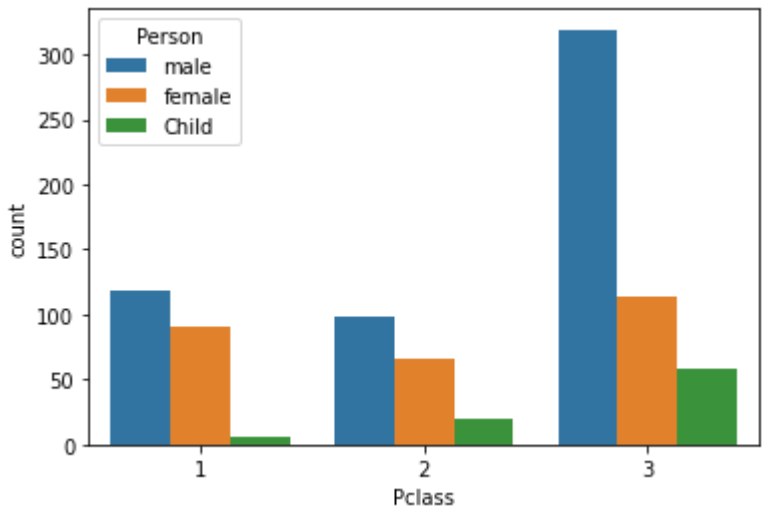
Out[24]: <AxesSubplot:ylabel='count'>



Bivariate Analytics

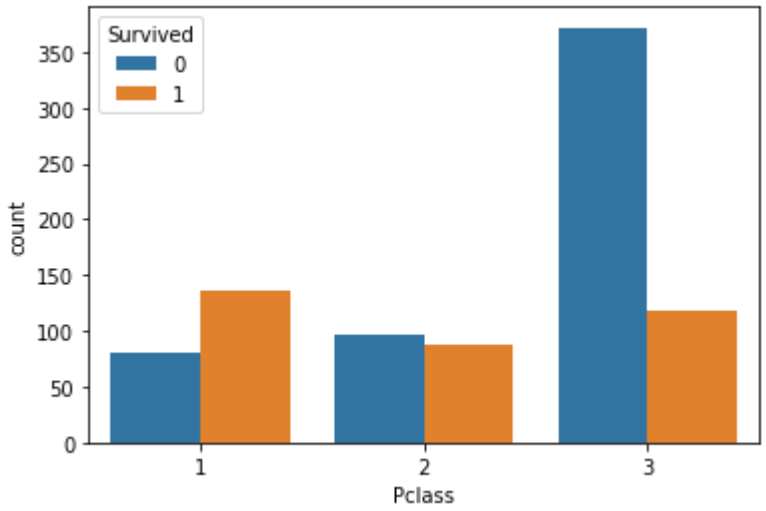
In [25]:
```
sns.countplot(x='Pclass', data=df_train,hue='Person')
```

Out[25]: <AxesSubplot:xlabel='Pclass', ylabel='count'>



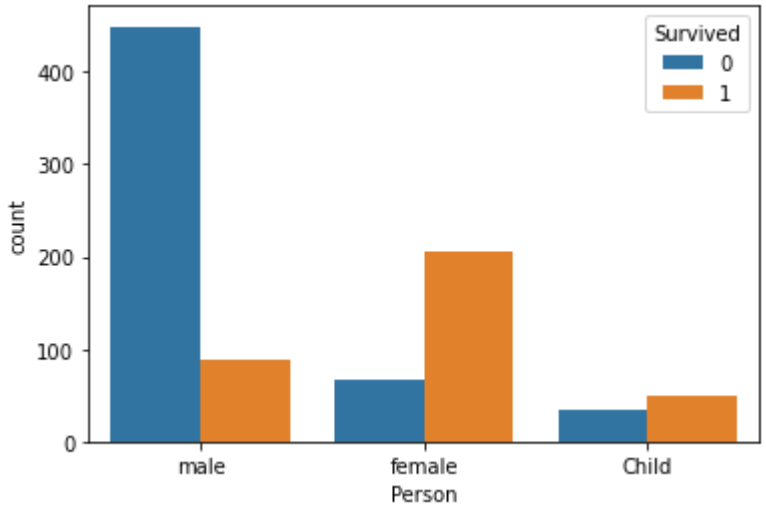
In [26]:
```
sns.countplot(x='Pclass', data=df_train,hue='Survived')
```

Out[26]: <AxesSubplot:xlabel='Pclass', ylabel='count'>



In [27]:
```
sns.countplot(x = 'Person',data = df_train,hue = 'Survived')
```

Out[27]: <AxesSubplot:xlabel='Person', ylabel='count'>

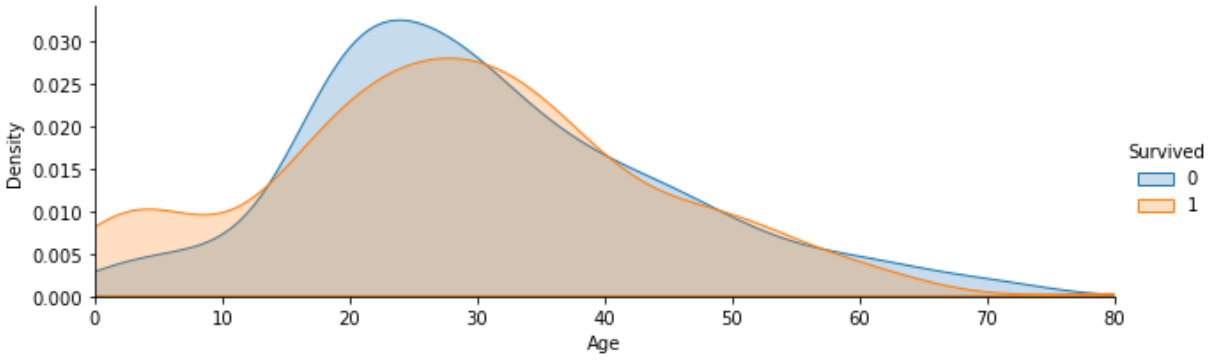


In [28]:
```
fig = sns.FacetGrid(df_train,aspect = 3,hue = "Survived")

fig.map(sns.kdeplot,'Age',shade = True)
oldest=df_train['Age'].max()

fig.set(xlim=(0, oldest))
fig.add_legend()
```

Out[28]:
```
<seaborn.axisgrid.FacetGrid at 0x21d4234ee20>
```

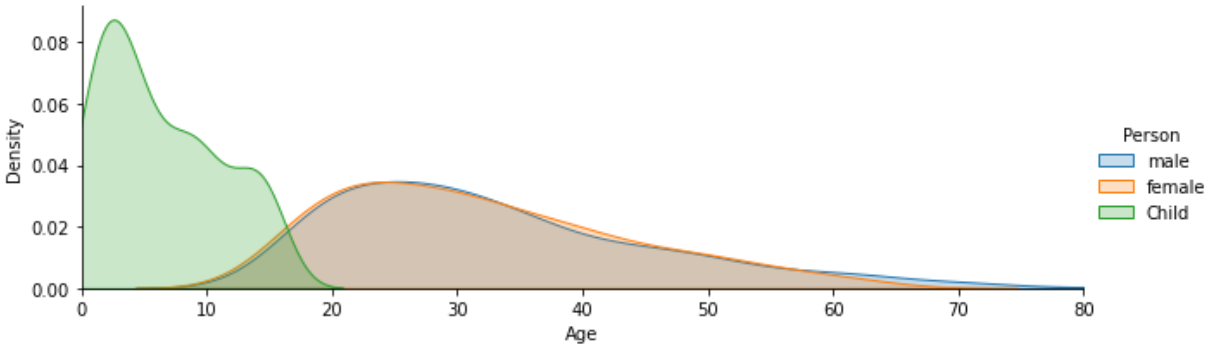


In [29]:
```
fig = sns.FacetGrid(df_train,aspect = 3,hue = "Person")

fig.map(sns.kdeplot,'Age',shade = True)
oldest=df_train['Age'].max()

fig.set(xlim=(0, oldest))
fig.add_legend()
```

Out[29]:
```
<seaborn.axisgrid.FacetGrid at 0x21d42348bb0>
```

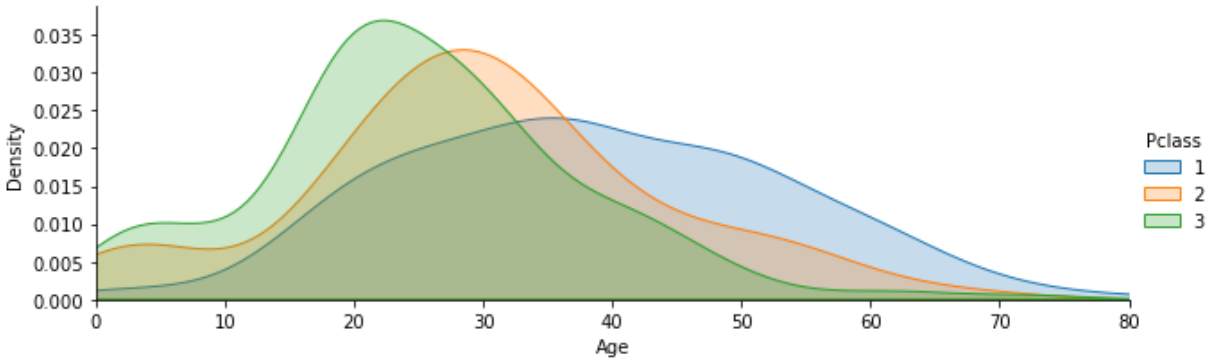


In [30]:
```
fig = sns.FacetGrid(df_train,aspect = 3,hue = "Pclass")

fig.map(sns.kdeplot,'Age',shade = True)
oldest=df_train['Age'].max()

fig.set(xlim=(0, oldest))
fig.add_legend()
```

Out[30]: <seaborn.axisgrid.FacetGrid at 0x21d4234e6a0>



In [31]:
```
df_train.head()
```

Out[31]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabi
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	Na
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C8
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	Na
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C12
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	Na

In [32]:
```
df_train['alone'] = df_train['SibSp'] + df_train['Parch']
```

In [33]:
```
df_train['alone']
```

Out[33]:
```
0      1
1      1
2      0
3      1
4      0
      ..
886    0
887    0
888    3
889    0
890    0
Name: alone, Length: 891, dtype: int64
```

In [34]:
```
df_train['alone'].loc[df_train['alone']>0] = "With Family"
df_train['alone'].loc[df_train['alone']==0] = "Alone"
```

```
C:\Users\HP\anaconda3\lib\site-packages\pandas\core\indexing.py:1732: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  self._setitem_single_block(indexer, value, name)
```

In [35]:
```
df_train
```

Out[35]:

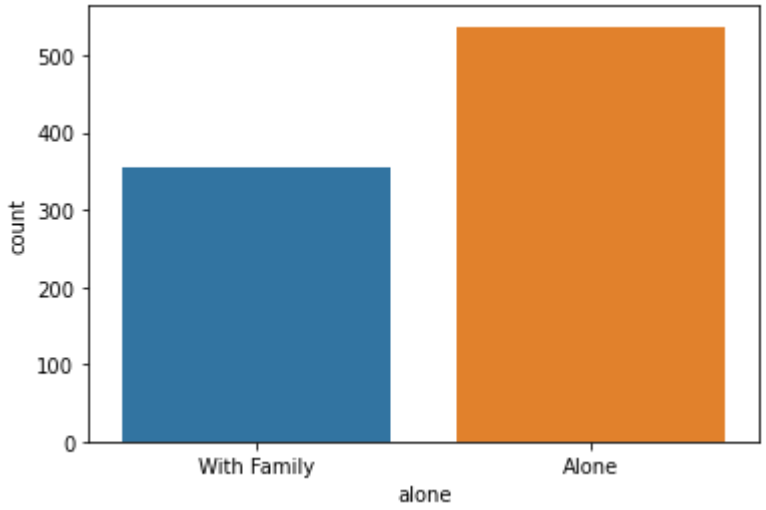
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Ca
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	
...	...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	

891 rows × 14 columns

In [36]:
```
sns.countplot('alone',data = df_train)
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarnin
g: Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without a
n explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

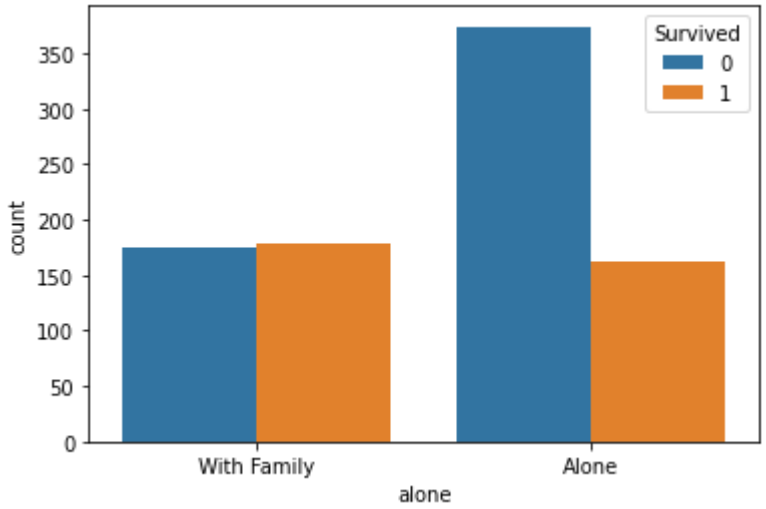
Out[36]:
```
<AxesSubplot:xlabel='alone', ylabel='count'>
```



In [37]:
```
sns.countplot('alone',data = df_train,hue = 'Survived')
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarnin
g: Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without a
n explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

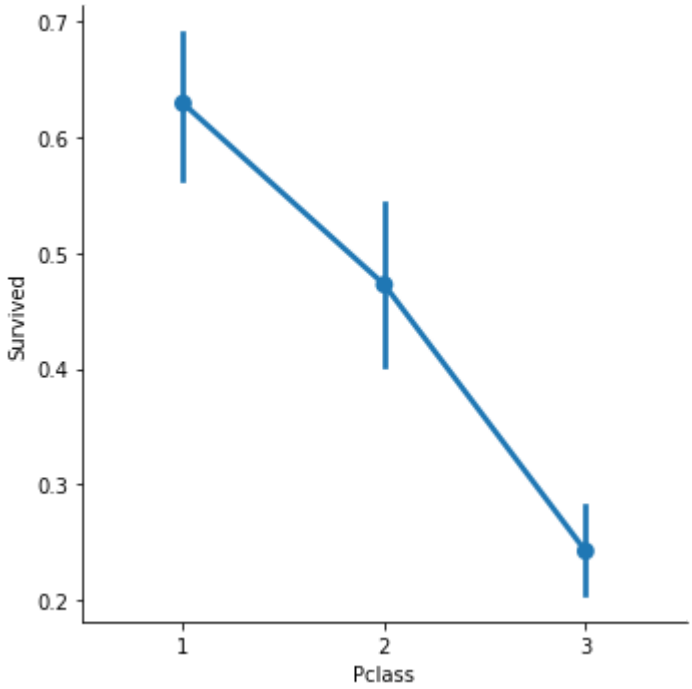
Out[37]:
```
<AxesSubplot:xlabel='alone', ylabel='count'>
```



In [38]:
```
sns.factorplot(x = 'Pclass',y = 'Survived',data = df_train)
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\categorical.py:3717: UserWarnin
g: The `factorplot` function has been renamed to `catplot`. The original name w
ill be removed in a future release. Please update your code. Note that the defa
ult `kind` in `factorplot` (`'point'`) has changed `'strip'` in `catplot`.
  warnings.warn(msg)
```

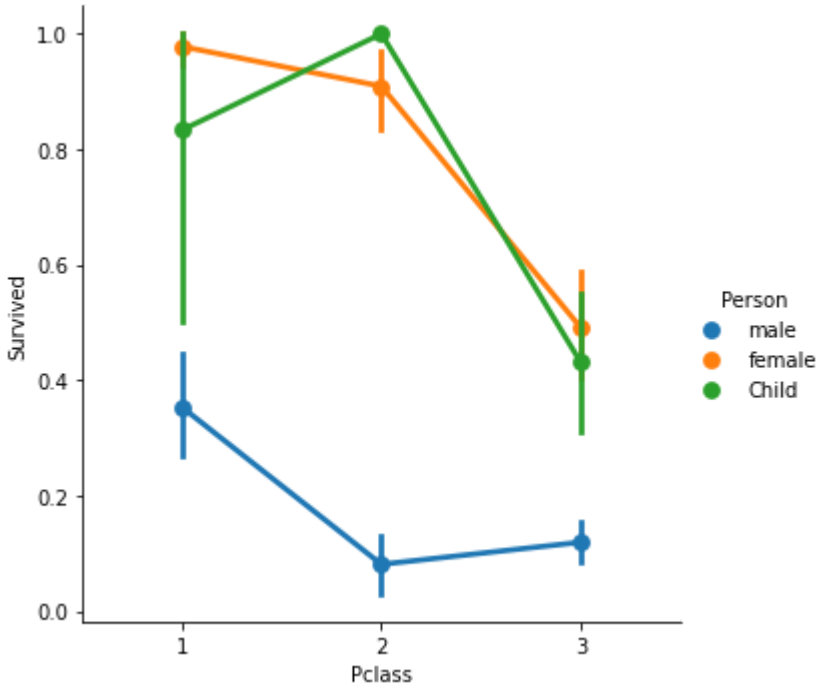
Out[38]:
```
<seaborn.axisgrid.FacetGrid at 0x21d423f4d90>
```



In [39]:
```
sns.factorplot(x = 'Pclass',y = 'Survived',data = df_train,hue = 'Person')
```

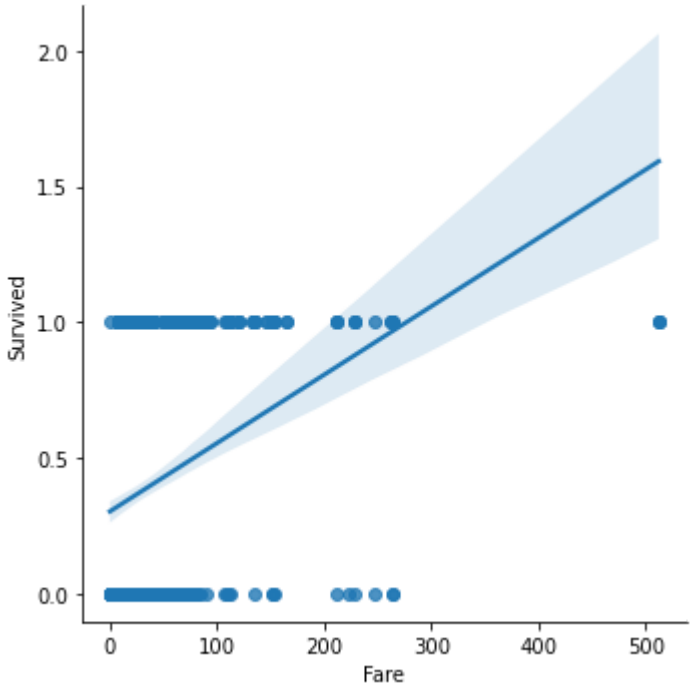
```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\categorical.py:3717: UserWarnin
g: The `factorplot` function has been renamed to `catplot`. The original name w
ill be removed in a future release. Please update your code. Note that the defa
ult `kind` in `factorplot` (`'point'`) has changed `'strip'` in `catplot`.
  warnings.warn(msg)
```

Out[39]: <seaborn.axisgrid.FacetGrid at 0x21d435a4160>



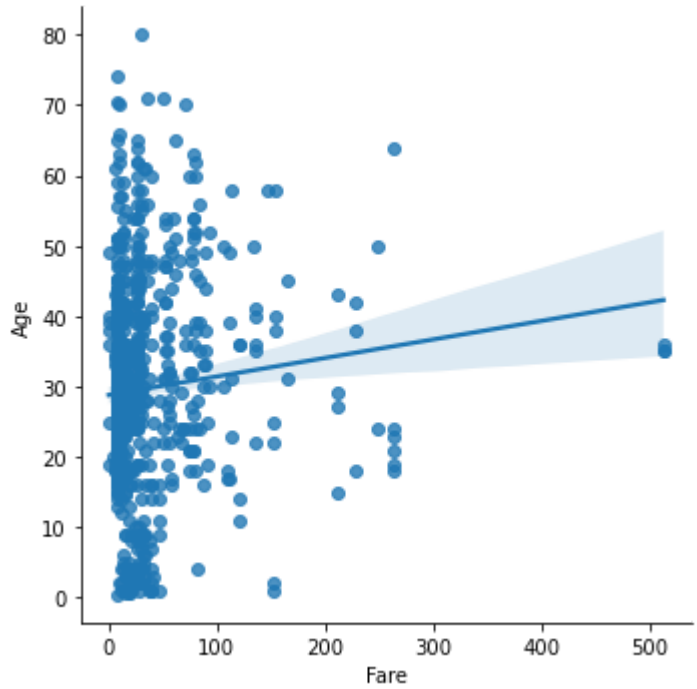
In [40]:
```
sns.lmplot(x = 'Fare',y = 'Survived',data = df_train)
```

Out[40]:
```
<seaborn.axisgrid.FacetGrid at 0x21d436fd1f0>
```



In [41]:
```
sns.lmplot(x = 'Fare',y = 'Age',data = df_train)
```

Out[41]:
```
<seaborn.axisgrid.FacetGrid at 0x21d4375e820>
```



In [42]:
```
df_train.head()
```

Out[42]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabi
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	Na
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C8
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	Na
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C12
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	Na

Steps to be followed:

1. Drop pid,Name,Ticket and cabin
2. Impute all other na value with mean or most occuring category
3. Perform categorical encoding to convert categories into number
4. Do a train test split with test size = 0.25 and random state = 123
5. Train and test the knn classifier and paste the f1 score in chat

In [43]:
```
df_train.drop(['PassengerId','Name','Ticket','Cabin'], axis=1,inplace=True) ## Dr
```

In [44]:
```
df_train
```

Out[44]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Person	alone
0	0	3	male	22.0	1	0	7.2500	S	male	With Family
1	1	1	female	38.0	1	0	71.2833	C	female	With Family
2	1	3	female	26.0	0	0	7.9250	S	female	Alone
3	1	1	female	35.0	1	0	53.1000	S	female	With Family
4	0	3	male	35.0	0	0	8.0500	S	male	Alone
...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S	male	Alone
887	1	1	female	19.0	0	0	30.0000	S	female	Alone
888	0	3	female	NaN	1	2	23.4500	S	female	With Family
889	1	1	male	26.0	0	0	30.0000	C	male	Alone
890	0	3	male	32.0	0	0	7.7500	Q	male	Alone

891 rows × 10 columns

In [45]:
```
#Impute all other na value with mean or most occuring category
df_train.isnull().sum()
```

Out[45]:
```
Survived      0
Pclass        0
Sex           0
Age         177
SibSp         0
Parch         0
Fare          0
Embarked      2
Person        0
alone         0
dtype: int64
```

In [46]:
```
df_train
```

Out[46]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Person	alone
0	0	3	male	22.0	1	0	7.2500	S	male	With Family
1	1	1	female	38.0	1	0	71.2833	C	female	With Family
2	1	3	female	26.0	0	0	7.9250	S	female	Alone
3	1	1	female	35.0	1	0	53.1000	S	female	With Family
4	0	3	male	35.0	0	0	8.0500	S	male	Alone
...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S	male	Alone
887	1	1	female	19.0	0	0	30.0000	S	female	Alone
888	0	3	female	NaN	1	2	23.4500	S	female	With Family
889	1	1	male	26.0	0	0	30.0000	C	male	Alone
890	0	3	male	32.0	0	0	7.7500	Q	male	Alone

891 rows × 10 columns

In [47]:
```
clean1_df_train=df_train.interpolate() #Impute NaN value with mean
```

In [48]:
```
clean1_df_train
```

Out[48]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Person	alone
0	0	3	male	22.0	1	0	7.2500	S	male	With Family
1	1	1	female	38.0	1	0	71.2833	C	female	With Family
2	1	3	female	26.0	0	0	7.9250	S	female	Alone
3	1	1	female	35.0	1	0	53.1000	S	female	With Family
4	0	3	male	35.0	0	0	8.0500	S	male	Alone
...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S	male	Alone
887	1	1	female	19.0	0	0	30.0000	S	female	Alone
888	0	3	female	22.5	1	2	23.4500	S	female	With Family
889	1	1	male	26.0	0	0	30.0000	C	male	Alone
890	0	3	male	32.0	0	0	7.7500	Q	male	Alone

891 rows × 10 columns

In [49]:
```
df_train = df_train. fillna(df_train['Age']. value_counts(). index[0])#Impute all
```

In [50]:
```
df_train
```

Out[50]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Person	alone
0	0	3	male	22.0	1	0	7.2500	S	male	With Family
1	1	1	female	38.0	1	0	71.2833	C	female	With Family
2	1	3	female	26.0	0	0	7.9250	S	female	Alone
3	1	1	female	35.0	1	0	53.1000	S	female	With Family
4	0	3	male	35.0	0	0	8.0500	S	male	Alone
...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	S	male	Alone
887	1	1	female	19.0	0	0	30.0000	S	female	Alone
888	0	3	female	24.0	1	2	23.4500	S	female	With Family
889	1	1	male	26.0	0	0	30.0000	C	male	Alone
890	0	3	male	32.0	0	0	7.7500	Q	male	Alone

891 rows × 10 columns

In [51]:
```
df_train = df_train. fillna(df_train['Embarked']. value_counts(). index[0])
```

In [52]:
```
##Perform categorical encoding to convert categories into number
df_train.dtypes
```

Out[52]:
```
Survived      int64
Pclass        int64
Sex          object
Age         float64
SibSp         int64
Parch         int64
Fare        float64
Embarked     object
Person       object
alone        object
dtype: object
```

In [53]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

In [54]:
```
df_train['Sex']=le.fit_transform(df_train['Sex'])
```

In [55]:
```
df_train
```

Out[55]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Person	alone
0	0	3	1	22.0	1	0	7.2500	S	male	With Family
1	1	1	0	38.0	1	0	71.2833	C	female	With Family
2	1	3	0	26.0	0	0	7.9250	S	female	Alone
3	1	1	0	35.0	1	0	53.1000	S	female	With Family
4	0	3	1	35.0	0	0	8.0500	S	male	Alone
...	...	...	...	...	...	...	...	...	...	...
886	0	2	1	27.0	0	0	13.0000	S	male	Alone
887	1	1	0	19.0	0	0	30.0000	S	female	Alone
888	0	3	0	24.0	1	2	23.4500	S	female	With Family
889	1	1	1	26.0	0	0	30.0000	C	male	Alone
890	0	3	1	32.0	0	0	7.7500	Q	male	Alone

891 rows × 10 columns

In [56]:
```
df_train['Person']=le.fit_transform(df_train['Person'])
```

In [57]:
```
df_train
```

Out[57]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Person	alone
0	0	3	1	22.0	1	0	7.2500	S	2	With Family
1	1	1	0	38.0	1	0	71.2833	C	1	With Family
2	1	3	0	26.0	0	0	7.9250	S	1	Alone
3	1	1	0	35.0	1	0	53.1000	S	1	With Family
4	0	3	1	35.0	0	0	8.0500	S	2	Alone
...	...	...	...	...	...	...	...	...	...	...
886	0	2	1	27.0	0	0	13.0000	S	2	Alone
887	1	1	0	19.0	0	0	30.0000	S	1	Alone
888	0	3	0	24.0	1	2	23.4500	S	1	With Family
889	1	1	1	26.0	0	0	30.0000	C	2	Alone
890	0	3	1	32.0	0	0	7.7500	Q	2	Alone

891 rows × 10 columns

In [58]:
```
df_train['alone']=le.fit_transform(df_train['alone'])
```

In [59]:
```
df_train
```

Out[59]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Person	alone
0	0	3	1	22.0	1	0	7.2500	S	2	1
1	1	1	0	38.0	1	0	71.2833	C	1	1
2	1	3	0	26.0	0	0	7.9250	S	1	0
3	1	1	0	35.0	1	0	53.1000	S	1	1
4	0	3	1	35.0	0	0	8.0500	S	2	0
...	...	...	...	...	...	...	...	...	...	...
886	0	2	1	27.0	0	0	13.0000	S	2	0
887	1	1	0	19.0	0	0	30.0000	S	1	0
888	0	3	0	24.0	1	2	23.4500	S	1	1
889	1	1	1	26.0	0	0	30.0000	C	2	0
890	0	3	1	32.0	0	0	7.7500	Q	2	0

891 rows × 10 columns

In [60]:
```
df_train.Embarked
```

Out[60]:
```
0      S
1      C
2      S
3      S
4      S
      ..
886    S
887    S
888    S
889    C
890    Q
Name: Embarked, Length: 891, dtype: object
```

In [61]:
```
df_train['Embarked'] = pd.to_numeric(df_train['Embarked'],errors = 'coerce')
```

In [62]:
```
df_train.dtypes
```

Out[62]:
```
Survived      int64
Pclass        int64
Sex           int32
Age         float64
SibSp         int64
Parch         int64
Fare        float64
Embarked    float64
Person        int32
alone         int32
dtype: object
```

In [63]:
```
df_train. drop(['Embarked'], axis=1, inplace=True)
```

In [64]:
```
df_train
```

Out[64]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Person	alone
0	0	3	1	22.0	1	0	7.2500	2	1
1	1	1	0	38.0	1	0	71.2833	1	1
2	1	3	0	26.0	0	0	7.9250	1	0
3	1	1	0	35.0	1	0	53.1000	1	1
4	0	3	1	35.0	0	0	8.0500	2	0
...	...	...	...	...	...	...	...	...	...
886	0	2	1	27.0	0	0	13.0000	2	0
887	1	1	0	19.0	0	0	30.0000	1	0
888	0	3	0	24.0	1	2	23.4500	1	1
889	1	1	1	26.0	0	0	30.0000	2	0
890	0	3	1	32.0	0	0	7.7500	2	0

891 rows × 9 columns

In [65]:
```
df_train.isnull().sum()
```

Out[65]:
```
Survived    0
Pclass      0
Sex         0
Age         0
SibSp       0
Parch       0
Fare        0
Person      0
alone       0
dtype: int64
```

In [66]:
```
#Do a train test split with test size = 0.25 and random state = 123

from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,roc_auc_score

df_train.describe()
```

Out[66]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Pe
count	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.00
mean	0.383838	2.308642	0.647587	28.566970	0.523008	0.381594	32.204208	1.50
std	0.486592	0.836071	0.477990	13.199572	1.102743	0.806057	49.693429	0.66
min	0.000000	1.000000	0.000000	0.420000	0.000000	0.000000	0.000000	0.00
25%	0.000000	2.000000	0.000000	22.000000	0.000000	0.000000	7.910400	1.00
50%	0.000000	3.000000	1.000000	24.000000	0.000000	0.000000	14.454200	2.00
75%	1.000000	3.000000	1.000000	35.000000	1.000000	0.000000	31.000000	2.00
max	1.000000	3.000000	1.000000	80.000000	8.000000	6.000000	512.329200	2.00

In [70]:
```
x = df_train.drop(['Survived'], axis=1)
y = df_train['Survived']
```

In [71]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=12
```

In [72]:
```
y_train.value_counts()
```

Out[72]:
```
0    410
1    258
Name: Survived, dtype: int64
```

In [73]:
```
#Train and test the knn classifier and paste the f1 score in chat
clf=KNeighborsClassifier(n_neighbors=5)
```

In [74]:
```
clf.fit(x_train,y_train)
```

Out[74]:
```
KNeighborsClassifier()
```

In [75]:
```
y_pred=clf.predict(x_test)
```

In [76]:
```
print(classification_report(y_pred,y_test))
```

```
              precision    recall  f1-score   support

           0       0.73      0.74      0.74       137
           1       0.58      0.57      0.58        86

    accuracy                           0.68       223
   macro avg       0.66      0.66      0.66       223
weighted avg       0.68      0.68      0.68       223
```

In [ ]:
```

```